

Разработка модуля визуализации данных при эвакуации экипажа сухогрузов навалного типа (балкеров) в Арктической зоне

Кондаков Максим Владимирович, 3 курс, ИСП-1 ПОЧУ «МКТ»

Бурзун Марина Сергеевна, преподаватель ПОЧУ «МКТ»

Актуальность обусловлена тем, что безопасность мореплавания является приоритетным направлением при оценке современного состояния морской транспортной системы и безопасности экипажа на судне

Цель исследования – разработка модуля визуализации данных при эвакуации экипажа сухогрузов навалного типа (балкеры) в Арктической зоне на основе Arduino IDE.

Задачи исследования:

1. Проанализировать статистику аварийности морских судов, связанной с экологическими катастрофами.
2. Выявить основные причины возникновения пожаров на судах приводящие к экологическим катастрофам.
3. Разработать модуль учета посещаемости и эвакуации экипажа сухогрузов навалного типа.
4. В исследовании широко применяются методы сравнения и обобщения.
5. Графическое описание выполнено с помощью структурно-логических схем и диаграмм.

Объектом исследования - является система оповещения и управления эвакуацией на морском судне.

Предмет исследования - подсистема учета информации о количестве членов экипажа, покинувших помещение.

Камеры для модуля учета людей, находящихся в помещении на морских судах



Камера HendPoplar на базе Arduino



Микросхема Micro SD D1 Mini Data для Arduino/Raspberry с камерой



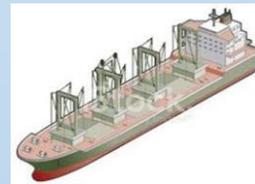
Модуль камеры OV7670 для Arduino

Диаграмма аварийных ситуаций на море

- повреждение корпуса и механизмов
- снос / посадка на мель
- пожары/взрывы
- аварии в результате смещения груза
- столкновение с судами
- столкновение с объектами
- затонувшие суда
- пропавшие суда



Теоретические основы исследования



1. Сухогруз навалного типа Балкеры предназначены для перевозки различных навалочных грузов, таких как руда, уголь, минеральные удобрения, стройматериалы или зерно, а также любой груз, который можно просто засыпать в трюмы без упаковки, то есть навалом или насыпью.

2. Морские эвакуационные системы – это комплекс спасательных средств, предназначенных для быстрой и эффективной эвакуации людей с палубы судна на спасательные шлюпки и плоты, расположенные на воде.



3. Визуализация данных — это представление данных в виде, который обеспечивает наиболее эффективную работу человека по их изучению.

Безопасность мореплавания



Разработка модуля учета людей, находящихся в помещении на морских судах

```

// Order C++ Compiler - Build, Compile and Run your C++ programs online in your favorite browser
#include "user/local/include/opencv2/opencv.hpp"
#include "user/local/include/opencv2/highgui.hpp"
#include "user/local/include/opencv2/imgproc.hpp"
#include "opencv.hpp"

using namespace cv;
using namespace std;

// Functions for Face Detection
void detectAndDisplay(const Mat& img, CascadeClassifier& classifier, CascadeClassifier& cascade, double scale);
void showFaces(const Mat& img, CascadeClassifier& classifier);

int main(int argc, char** argv) {
    // VideoCapture class for capturing video from which faces are detected
    VideoCapture cap;
    Mat frame;
    // Predefined named XML classifiers with facial features
    CascadeClassifier cascade(classifier);
    double scale=1;
    // Load classifier from 'opencv_data/haarcascade_frontalface_alt.xml'
    CascadeClassifier classifier(classifier);
    // Change path before execution
    cascade.load("opencv_data/haarcascade_frontalface_alt.xml");
    // Open Video_11 for the WebCam? Path to Video_11 for a Local Video
    cap.open(argv[1]);
    // Capture frames from video and detect faces
    cap << "Face Detection Started." << endl;
    while(1) {
        // Capture frame from video
        cap >> frame;
        // Detect faces from the frame
        detectAndDisplay(frame, classifier, cascade, scale);
        // Press 'q' to quit the program
        if (cvWaitKey(1) <= 'q') break;
    }
    cap.release();
    return 0;
}
    
```

```

void detectAndDisplay(const Mat& img, CascadeClassifier& classifier, CascadeClassifier& cascade, double scale) {
    // Detect faces from the image
    vector<Rect_> faces;
    classifier.detectMultiScale(img, faces, scale, 1.5, 1, 0, 0);
    // Draw circles around the faces
    for (int i=0; i<faces.size(); i++) {
        Rect_ face = faces[i];
        int cx = face.x + face.width/2;
        int cy = face.y + face.height/2;
        int radius = cvRound((face.width + face.height)/2.5);
        circle(img, Point_((cx), (cy)), radius, Scalar_((255), (0), (0)), 2);
    }
    // Show the processed image with detected faces
    imshow("Processed Image with detected faces", img);
    // Wait for a key to be pressed
    cvWaitKey(1);
}
    
```

```

// Detection of eyes in the input image
nestedCascade_detectSubScale(const Mat_& imgROI, nestedObjects_1, 1, 2, CascadeClassifier_& cascade, int_& size, int_& scale);

// Draw circles around eyes
for (int i=0; i<nestedObjects_1.size(); i++) {
    Rect_ eye = nestedObjects_1[i];
    center_x = cvRound((eye.x + eye.width/2)*scale);
    center_y = cvRound((eye.y + eye.height/2)*scale);
    radius = cvRound((eye.width + eye.height)/2.5)*scale;
    circle(img, center, radius, color, 3, 8, 0);
}

// Show Processed Image with detected faces
imshow("Processed Image with detected faces", img);
cvWaitKey(1);
}
    
```

